FIG. 1

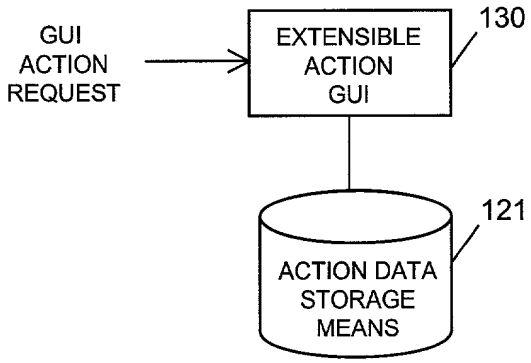

FIG. 2

```
1  /* simplified algorithm for a Collection Extensible Action GUI */
2  Receive a request to execute a GUI action
3  Obtain action definition from action data storage means
4  Execute the requested action according to stored action data
5  Display execution results on display screens, in computer files, etc.
```

FIG. 3

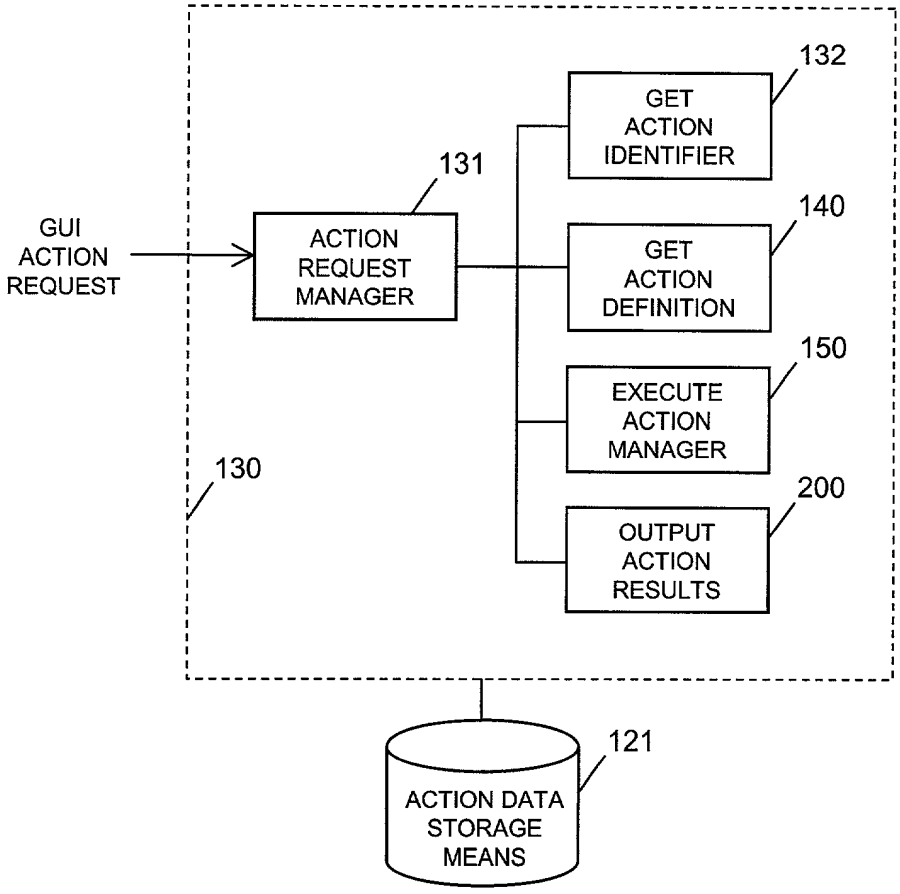

FIG. 4

```
1  /* simplified algorithm for a Module Action Request Manager */
2  Receive action execution request (e.g. from menu or button click)
3  Get action identifier from incoming action execution request
4  Get action definition from executable action data storage means
5  Execute requested action according to executable action definition
6  Output action execution results on display screens, etc.
```

FIG. 5

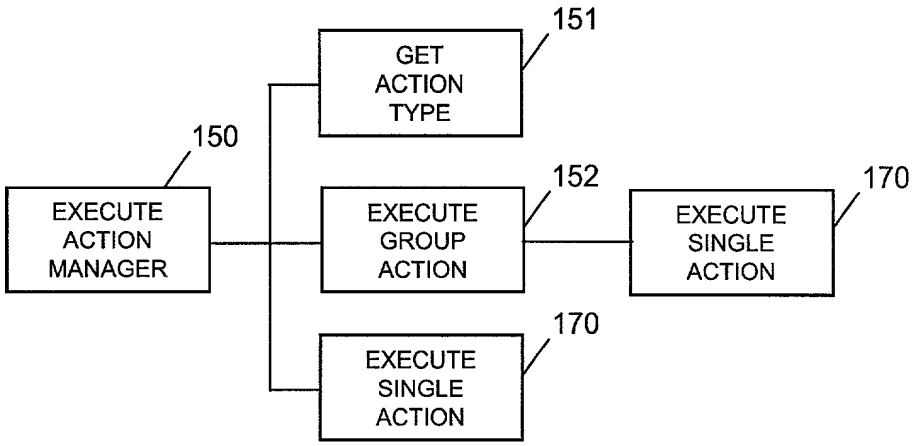

FIG. 6

```
1   /* simplified algorithm for Module Execute Action Manager */
2   Determine requested action type (single or group action)
3   If requested action is a group action, call Execute Group Action
4       - Loop to execute group action, calling Execute Single Action
5   If requested action is a single action, call Execute Single Action
```

FIG. 7

```
1   /* simplified algorithm for Module Execute Group Action */
2   Loop over all single actions in the group
3       Call Execute Single Action for each action in turn
```

FIG. 8

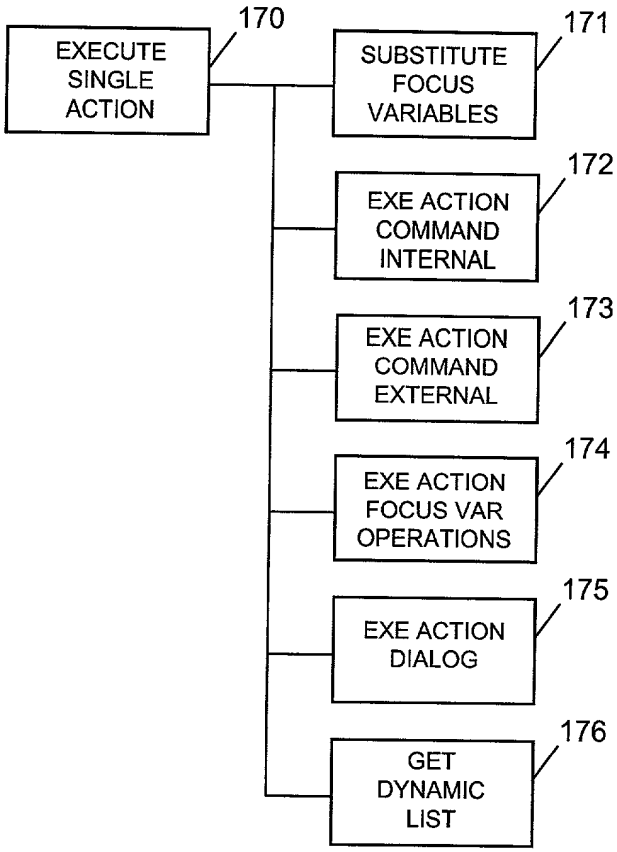

FIG. 9

```
1  /* simplified algorithm for Module Execute Single Action */
2  Substitute focus variables into action command templates
3  If action type = internal, call Execute Action Command Internal
4  If action type = external, call Execute Action Command External
5  If action type = fvar-op, call Execute Action Focus Variable Operations
6  Calculate dynamic list if required by a dialog
7  If action type = dialog, call Execute Action Dialog
```

FIG. 10

```
1  /* GUI action request data structure */
2  request-info {

3    + action-identifier

4    + other optional request information, if desired
5  }
```

FIG. 11

```
1   name-action.tbl:
2   # action names and their definition files
3   # name                    definition-file
4   #
5   a-coll-file-edit          a-coll.def
6   a-coll-file-new           a-coll.def
7   a-coll-file-open          a-coll.def
8   a-coll-file-save          a-coll.def
9   a-build-and-export        a-action-misc.def
10  ...
11  a-focus-coll              a-focus-coll.def
12  a-focus-repository        a-focus-repository.def
13  a-focus-prev-role         a-focus-prev-role.def
14  ...
15  a-display-message         a-action-misc.def
16  a-file-cmd-dir            a-action-misc.def
17  a-edit-find-and-replace   a-action-misc.def
18  ...
19  a-filename-set            a-action-vars.def
20  a-pathname-set            a-action-vars.def
21  a-coll-select             a-coll.def
22  a-coll-build-and-install  a-coll.def
```

FIG. 12

```
1   # The format of an action definition is shown below:
2   #
3   # action                <action name>
4   # action-desc           <one line text description>
5   # action-type           <action type = SINGLE, GROUP>
6   #
7   # action-cmd-aname      <action name for group actions>
8   # action-cmd-dname      <dialog name>
9   # action-cmd-iname      <internal subroutine name>
10  # action-cmd-xtext      <external cmd text>
11  #
12  # action-pcmd-xtext     <execute a command in parallel>
13  # action-pcmd-aname     <execute a single action in parallel>
14  # action-pcmd-wait      <wait for prev parallel exec block to finish>
15  #
16  # action-var-dirname    <left side (dirname) of pathname>
17  # action-var-filename   <right side (filename) of pathname>
18  # action-var-varname    <variable name>
19  #
20  # action-ifile          <input  filename>
21  # action-odisplay       <display method = popup/galley/none>
22  # action-ofile          <output filename>
23  # action-ofile-save     <save output file  - yes/no>
24  #
25  # end-action
```

FIG. 13

```
1   a-action-misc.def:
2   # some example actions and their implementations
3   #
4   action                  a-display-message
5   action-desc             display a message in a text box
6   action-type             single
7   action-cmd-iname        callback-text-display
8   action-cmd-text         “Creating customized files now...”
9   end-action
10  #
11  action                  a-file-cmd-dir
12  action-desc             run the “dir” command and display output
13  action-type             single
14  action-cmd-iname        callback-xcmd
15  action-cmd-xtext        "command.com /c dir"
16  action-odisplay         win-galley
17  end-action
18  #
19  action                  a-edit-find-and-replace
20  action-desc             run a program to find and replace strings
21  action-type             single
22  action-cmd-iname        callback-fvar
23  action-cmd-dname        dialog-find-and-replace
24  action-cmd-text         “changeme “@{old}” “@{new}” –files “@{files}”
25  end-action
26  #
27  action                  a-build-and-export
28  action-desc             generate makefile, build, and export a program
29  action-type             single
30  action-cmd-iname        callback-platform
31  action-cmd-xtext        “makefilemagic –outfile makefile”
32  action-cmd-xtext        “make all”
33  action-cmd-xtext        “make exports”
34  action-cmd-xtext        “make clean”
35  #
36  end-action
```

FIG. 14

```
a-action-vars.def:
# some example actions that work with variables
#
action               a-pathname-set
action-desc          put user pathname into a variable
action-type          single
action-cmd-iname     callback-var-set
action-var-varname   var-user-pathname
action-var-value     c:/some/dirname
end-action
#
action               a-filename-split
action-desc          split pathname into dir and filename parts
action-type          single
action-cmd-iname     callback-var-split
action-var-varname   var-user-pathname
action-var-dirname   var-user-dirname
action-var-filename  var-user-filename
#
end-action
```

FIG. 15

```
1   # some example focus variable names and their values
2   #   name              substitution-value
3   #
4   cfg-cust-ctx          default
5   cfg-cust-dir-cust     c:\codefast\version\site
6   cfg-cust-dir-inst     c:\codefast
7   cfg-cust-inst-type    cf-both
8   cfg-cust-platform     win2000.plt
9   cfg-cust-tree         NEW
10  fgui-context          default
11  fgui-base-dir         _HOME_\code
12  fgui-collection       testme
13  fgui-role             developer
14  fgui-timeset          default
15  prev-coll             c-myprogram
16  old                   my-old-string
17  new                   my-new-string
18  files                 "filename1 filename2 ... filename3.txt"
```

FIG. 16

```
1   name-dialog.tbl:
2   # dialogs and their definition files
3   #
4   d-coll-select              d-coll-select.def
5   d-coll-rename              d-coll-rename.def
6   d-find-and-replace         d-find-and-replace.def
```

FIG. 17

```
1   d-coll-select.def:
2   # dialog for selecting a collection from a list
3   #
4   dialog                     d-coll-select
5   dialog-title               Select Collection
6   #
7   # define a text box to store coll name in a variable
8   textbox                    coll-name
9   textbox-title              "Select A Collection"
10  #
11  # generate a dynamic list of colls to select from
12  textbox-list-type          list-dynamic
13  textbox-list-xcmd          "listcolls –r"
14  textbox-list-label         "List of collections to select from:"
15  #
16  # a non-null answer is required from the user
17  textbox-text-required      yes
18  #
19  # store the selection result in this variable name
20  textbox-var-name           var-selected-coll
21  #
22  ... other dialog attributes as desired
23  end-dialog
```

FIG. 18

```
1   name-list.tbl:
2   # example lists and their definition files
3   #
4   list-projects              list-projects.def
5   list-personnel             list-personnel.def

6   list-projects.def:
7   # list of projects at this site
8   # display-name             return-value
9   #
10  "Admin Systems"            proj-admin-sys
11  "Network Upgrade"          proj-net-upgrade

12  list-personnel.def:
13  # list of employees
14  # display-name             return-value
15  Tom                        tom
16  Joe                        joe
17  Susan                      susan
```

FIG. 19

```
1   a-coll-select-and-install.def:
2   # action def for selecting a coll, building it, and installing it
3   #
4   action                  a-coll-select-and-install
5   action-desc             select, build, and install a collection
6   action-type             group
7   action-cmd-iname        callback-mult-commands
8   action-cmd-aname        a-coll-select
9   action-cmd-aname        a-coll-build-and-install
10  end-action
```

FIG. 20

```
1   a-coll.def:
2   # action definitions for working with collections
3   #
4   action                  a-coll-select
5   action-desc             select a collection from a list of collections
6   action-type             single
7   action-iname            callback-collection-open
8   end-action
9   #
10  action                  a-coll-build-and-install
11  action-desc             create makefile, build, and install a program
12  action-type             single
13  action-cmd-iname        callback-platform
14  action-cmd-xtext        “makefilemagic –outfile makefile”
15  action-cmd-xtext        “make all”
16  action-cmd-xtext        “make install”
17  end-action
```

FIG. 21

```
1   a-coll.def:
2   # an example parallel single action with parallel external commands
3   #
4   action                  a-single-parallel
5   action-desc             run three makefile targets in parallel
6   action-type             single
7   action-cmd-iname        callback-platform
8   action-pcmd-xtext       “make target-one”
9   action-pcmd-xtext       “make target-two”
10  action-pcmd-xtext       “make target-three”
11  end-action
```

FIG. 22

```
1   a-coll.def:
2   # an example parallel single action with a mixed combination
3   # of sequential and parallel external commands
4   #
5   action                  a-single-parallel
6   action-desc             run three makefile targets in parallel
7   action-type             single
8   action-cmd-iname        callback-platform
9   #
10  # execute this command by itself
11  action-cmd-xtext        “command 1”
12  #
13  # execute the next two commands in parallel
14  action-pcmd-xtext       “command 2”
15  action-pcmd-xtext       “command 3”
16  #
17  # execute the next command by itself
18  action-cmd-xtext        “command 4”
19  #
20  end-action
```

FIG. 23

```
1   a-coll.def:
2   # an example parallel single action with a sequence of
3   # parallel execution blocks separated by a wait command
4   #
5   action                  a-single-parallel
6   action-desc             run 2 blocks of 2 commands in parallel
7   action-type             single
8   action-cmd-iname        callback-platform
9   #
10  # execute the next two commands in parallel
11  action-pcmd-xtext       “command 2”
12  action-pcmd-xtext       “command 3”
13  #
14  # wait for the previous execution block to finish
15  action-pcmd-wait
16  #
17  # execute the next two commands in parallel
18  action-pcmd-xtext       “command 2”
19  action-pcmd-xtext       “command 3”
20  #
21  end-action
```

FIG. 24

```
1   a-actions-parallel.def:
2   # an example parallel group action for building multiple programs
3   #
4   action                  a-build-group-one-programs
5   action-desc             build several programs in parallel
6   action-type             group
7   action-cmd-iname        callback-mult-commands
8   action-pcmd-aname       a-build-group-one
9   action-pcmd-aname       a-build-group-two
10  end-action
```

FIG. 25

```
1   a-actions-parallel.def:
2   # an example parallel group action containing an interleaved sequence
3   # of single actions to be executed in sequence and in parallel
4   #
5   action                  a-build-group-one-programs
6   action-desc             build several programs in parallel
7   action-type             group
8   action-cmd-iname        callback-mult-commands
9   #
10  # execute this single action by itself
11  action-cmd-aname        a-single-action-1
12  #
13  # execute the next two single actions in parallel
14  action-pcmd-aname       a-single-action-2
15  action-pcmd-aname       a-single-action-3
16  #
17  # execute the next single action by itself
18  action-cmd-aname        a-single-action-4
19  #
20  end-action
```

FIG. 26

```
1   a-actions-parallel.def:
2   # an example parallel group action containing a sequence
3   # of parallel execution blocks separated by a wait command
4   #
5   action                      a-build-group-one-programs
6   action-desc                 build several programs in parallel
7   action-type                 group
8   action-cmd-iname            callback-mult-commands
9   #
10  # execute the next two single actions in parallel
11  action-pcmd-aname           a-single-action-2
12  action-pcmd-aname           a-single-action-3
13  #
14  # wait for the previous parallel execution block to finish
15  action-pcmd-wait
16  #
17  # execute the next two single actions in parallel
18  action-pcmd-aname           a-single-action-2
19  action-pcmd-aname           a-single-action-3
20  #
21  end-action
```